

```
import { useDispatch } from 'react-redux';
```

→ Redux से useDispatch को इम्पोर्ट किया है।

🧠 सोचिए आप दुकान में सेल्समैन को बोलते हैं "ये सामान पैक कर दो" — dispatch वही काम करता है, एक्शन भेजता है।

js

Copy

Edit

```
import { addToCart } from '../features/cartSlice';
```

→ Redux के cartSlice से "addToCart" एक्शन को लाया गया है।

🧠 जैसे आप शॉपिंग में कोई चीज टोकरी में डालते हैं — ये वही काम करता है, item को cart में डालता है।

js

Copy

Edit

```
import { useNavigate } from 'react-router-dom';
```

→ React Router से navigation का hook लाया गया है।

🧠 जैसे आप Google Maps में रास्ता बदलते हैं — useNavigate वेबसाइट में पेज बदलने का रास्ता देता है।

js

Copy

Edit

```
import "../style/FashionPage.css";
```

→ CSS फाइल जो स्टाइल देती है, उसे इम्पोर्ट किया गया है।

🧠 जैसे कोई कपड़े को पहनने के बाद सजाया जाता है – CSS हमारे पेज को खूबसूरत बनाता है।

js

Copy

Edit

```
import fashionImages from "../data/fashionData";
```

→ fashion product की लिस्ट डाटा फाइल से लाया गया है।

🧠 जैसे ऑनलाइन शॉपिंग साइट में सारे कपड़ों की एक लिस्ट होती है — वैसे ही ये है।

js

Copy

Edit

```
const FashionPage = ({ searchTerm }) => {
```

→ FashionPage नाम का component शुरू होता है, जो searchTerm को props में लेता है।

🧠 मान लो यूज़र ने search box में "jeans" लिखा — वो searchTerm यहाँ आ रहा है।

js

Copy

Edit

```
const [sortType, setSortType] = useState("");
```

→ sortType नाम का state बनाया गया है, जो बताएगा प्राइस कम या ज़्यादा के हिसाब से सॉर्ट करना है।

🧠 जैसे Flipkart में "Price low to high" या "high to low" ऑप्शन होता है।

js

Copy

Edit

```
const dispatch = useDispatch();
```

→ Redux के action भेजने के लिए dispatch फंक्शन बनाया गया।

🧠 जैसे आप वेटर को ऑर्डर बोलते हैं — dispatch वैसा ही है।

js

Copy

Edit

```
const navigate = useNavigate();
```

→ navigate फंक्शन बनाया गया है जिससे हम किसी और पेज पर जा सकें।

🧠 जैसे आप एक शॉप से दूसरी शॉप जाते हैं — navigate वही करता है वेबसाइट पर।

js

Copy

Edit

```
const handleAddToCart = (product) => {  
  dispatch(addToCart(product));  
};
```

→ यह फंक्शन product को cart में डालता है।

🧠 जैसे आपने कोई ड्रेस पसंद की और "Add to cart" दबाया — वो काम यह लाइन करती है।

js

Copy

Edit

```
const handleProductClick = (item) => {  
  console.log("Navigating to product with ID:", item.id);  
  navigate(`/fashion-product/${item.id}`);  
};
```

→ यह function उस प्रोडक्ट की डिटेल पेज पर ले जाता है जब यूज़र उसे क्लिक करता है।

🧠 जैसे आप Myntra में किसी कपड़े पर क्लिक करते हैं तो उसकी details खुलती है।

js

Copy

Edit

```
let filtered = Array.isArray(fashionImages) ? fashionImages.filter((item) =>  
  item.title?.toLowerCase().includes((searchTerm || "").toLowerCase())  
) : [];
```

→ यह searchTerm के हिसाब से product को filter करता है।

🧠 मान लीजिए आपने "shirt" सर्च किया — तो सिर्फ वही प्रोडक्ट आएंगे जिनके नाम में "shirt" है।

js

Copy

Edit

```
if (sortType === "low") {  
  filtered.sort((a, b) => a.price - b.price);  
} else if (sortType === "high") {
```

```
filtered.sort((a, b) => b.price - a.price);
}
```

→ यह प्राइस के हिसाब से list को sort करता है।

🧠 जैसे आप Flipkart में प्राइस कम से ज़्यादा दिखाते हैं।

js

Copy

Edit

```
return (
  <div className="fashion-page">
```

→ यह पूरी फैशन पेज का HTML रेंडर करना शुरू करता है।

js

Copy

Edit

```
    <div className="sort-buttons">
      <button onClick={() => setSortType("low")}>Price: Low to High</button>
      <button onClick={() => setSortType("high")}>Price: High to Low</button>
    </div>
```

→ यह दो बटन बनाता है जिनसे यूज़र प्राइस के आधार पर sort कर सकता है।

🧠 मान लीजिए यूज़र कहता है, "मुझे सस्ते वाले पहले दिखाओ।"

js

Copy

Edit

```
    <div className="fashion-product-grid">
      {filtered.map((item, idx) => (
```

→ यह सभी filtered products को लूप में दिखाना शुरू करता है।

js

Copy

Edit

```
        <div
          className="fashion-product-card"
          key={item.id || idx}
          onClick={() => handleProductClick(item)}
        >
```

→ हर प्रोडक्ट को एक card में दिखाता है और उस पर क्लिक करने पर detail page खोलता है।

js

Copy

Edit

```
        <div className="fashion-product-image-container">
          <img src={item.img} alt={item.title} />
```

→ प्रोडक्ट की फोटो दिखाता है।

js

Copy

Edit

```
<div className="fashion-hover-overlay">
  <h3>{item.title}</h3>
  <p>₹{item.price}</p>
```

→ जब आप hover करते हैं तो टाइटल और प्राइस दिखता है।

js

Copy

Edit

```
<button
  className="fashion-add-to-cart-btn"
  onClick={(e) => {
    e.stopPropagation();
    handleAddToCart(item);
  }}
>
  Add to Cart
</button>
```

→ Add to Cart बटन है, जो क्लिक करने पर item cart में डालता है और detail page पर नहीं जाता।

js

Copy

Edit

```
    </div>
  </div>
</div>
  )))
</div>
</div>
);
};
```

→ सभी HTML structure को बंद कर दिया गया और पूरा पेज दिखा दिया गया।

js

Copy

Edit

```
export default FashionPage;
```

→ इस component को बाहर एक्सपोर्ट किया ताकि कोई और फाइल इसे इस्तेमाल कर सके।

अगर चाहो तो मैं इसका एक छोटा mockup भी बना सकता हूँ या फिर sorting/filtering को और interactive कर सकता हूँ! 😊

e.stopPropagation() क्या करता है?

→ यह एक JavaScript method है जो **event** को ऊपर के **parent elements** तक जाने से रोकता है।

Imagine Example (कल्पना कीजिए):

आपने एक प्रोडक्ट कार्ड पर क्लिक किया — पूरा कार्ड क्लिक करने से प्रोडक्ट डिटेल पेज खुल जाता है।

अब उस कार्ड के ऊपर एक "Add to Cart" बटन भी है।

बिना `e.stopPropagation()`:

अगर आप सिर्फ "Add to Cart" बटन पर क्लिक करते हो, तो:

1. वह item cart में तो जाएगा ✅
2. लेकिन card का outer click भी trigger हो जाएगा 😞
3. और आप product detail page पर भी पहुँच जाओगे — जो हम नहीं चाहते!

लेकिन अगर आप लिखते हो:

js

CopyEdit

```
e.stopPropagation();
```

तो ये कहता है:

"ये click सिर्फ इस बटन तक ही सीमित रहेगा, ऊपर वाले कार्ड को मत बताना कि click हुआ है।"

यानि: ✅ Item cart में जाएगा

❌ लेकिन डिटेल पेज पर नहीं जाएगा

आसान भाषा में:


"बटन बोलता है — ओ भाई, ये click मेरा है, card को मत बताओ।"

js

CopyEdit

```
import React, { useState, useEffect } from 'react';
```

React, useState और useEffect को इम्पोर्ट किया गया।

 सोचो `useState` = दिमाग की याददाश्त, `useEffect` = जैसे अलार्म सेट करना - जब कुछ बदले तब काम करो।

js

CopyEdit

```
import '../style/header.css';
```

→ CSS फाइल से header का लुक लाया गया है।

👉 जैसे कपड़े पहनकर इंसान अच्छा दिखता है, वैसे ही CSS से component सजता है।

js

CopyEdit

```
import { FaLocationDot, FaMagnifyingGlass, FaCartShopping } from  
'react-icons/fa6';
```

→ Icons को react-icons लाइब्रेरी से लाया गया है।

👉 जैसे मोबाइल ऐप में छोटे-छोटे symbols होते हैं— location, search, cart वगैरह।

js

CopyEdit

```
import { Link, useNavigate } from 'react-router-dom';
```

→ React Router से Link और Navigate फंक्शन लिए गए हैं।

👉 Link = रास्ता, navigate = किसी जगह पर ले जाना (जैसे Google Maps)।

js

CopyEdit

```
import allProducts from '../data/allProducts';
```

→ सभी प्रोडक्ट्स का data एक file से लाया गया है।

👉 जैसे स्टोर में सभी items की लिस्ट होती है — वैसे यहाँ।

js

CopyEdit

```
function Header({ searchTerm, setSearchTerm, onSearch }) {
```

→ Header component शुरू होता है, जिसमें props से सर्च टर्म और फंक्शन आते हैं।

👉 यूज़र जो टाइप करता है — वो searchTerm है।

js

CopyEdit

```
const [localSearch, setLocalSearch] = useState(searchTerm);
```

➡ **Local input की value को state में रखा गया है।**

🧠 जैसे टाइप करते-करते अपने मन में शब्द याद रखते हो।

js

CopyEdit

```
const [suggestions, setSuggestions] = useState([]);
```

➡ **Auto-suggestions दिखाने के लिए खाली array बनाया गया।**

💡 जैसे Google search करते समय नीचे suggestion आता है — वैसा ही।

js

CopyEdit

```
const [selectedCategory, setSelectedCategory] = useState('All');
```

➡ **ड्रॉपडाउन से चुनी गई category की state है।**

🛒 जैसे Flipkart में "Mobile", "Fashion", "All" से चुनते हो।

js

CopyEdit

```
const navigate = useNavigate();
```

➡ **Navigation के लिए function बनाया जो पेज बदल सके।**

🚗 जैसे क्लिक करने पर नए पेज पर ले जाना।

🔄 **useEffect - Auto Suggestion का दिमाग**

js

CopyEdit

```
useEffect(() => {  
  if (localSearch.trim() === '') {  
    setSuggestions([]);  
    return;  
  }  
})
```

➡ **अगर यूज़र ने कुछ नहीं लिखा है, तो suggestions हटा दो।**

js

CopyEdit

```
const filtered = allProducts.filter(item =>
  item.title?.toLowerCase().includes(localSearch.toLowerCase())
  &&
  (selectedCategory === 'All' || item.type?.toLowerCase() ===
    selectedCategory.toLowerCase())
  );
```

➡ **searchTerm और category के हिसाब से products को filter किया।**

🔍 जैसे आपने “Samsung” सर्च किया और category में “Mobile” चुना।

js

CopyEdit

```
setSuggestions(filtered.slice(0, 5));
}, [localSearch, selectedCategory]);
```

➡ **Filtered items में से केवल टॉप 5 suggestions दिखाए गए।**

🔥 Input Handler

js

CopyEdit

```
const handleInputChange = (e) => {
  setLocalSearch(e.target.value);
  setSearchTerm(e.target.value);
  onSearch(e);
};
```

➡ **जैसे-जैसे यूज़र टाइप करता है, state अपडेट होती है।**

✅ Suggestion पर क्लिक

js

CopyEdit

```
const handleSuggestionClick = (title) => {
  setLocalSearch(title);
  setSuggestions([]);
};
```



```
navigate(`/search-results?query=${title}&category=${selectedCategory}
`));
};
```

➡ सजेशन पर क्लिक करने से यूज़र को सर्च रिजल्ट वाले पेज पर ले जाया जाता है।

Search बटन पर क्लिक

js

CopyEdit

```
const handleSearchClick = () => {
  if (localSearch.trim() !== '') {

navigate(`/search-results?query=${localSearch}&category=${selectedCa
tegory}`);
  }
};
```

➡ अगर सर्च बॉक्स खाली नहीं है तो सर्च रिजल्ट पेज पर redirect किया जाता है।

Header Layout शुरू

js

CopyEdit

```
return (
  <div className="navbar">
```

➡ पूरा navbar यहीं से शुरू होता है।


Logo और Location

js

CopyEdit

```
<div className="nav-logo border">...</div>
<div className="nav-address border">
  <p className="add-first">Deliver to</p>
  <div className="add-icon">
    <FaLocationDot />
```

```
        <p className="add-second">India</p>
      </div>
    </div>
```


 Logo और India location दिखाई जाती है — जैसे Amazon पर होता है।

Search Box + Dropdown

js

CopyEdit

```
<div className="nav-search">...</div>
```

 Dropdown, input, और search icon सब एक साथ मिलाकर search bar बनता है।

js

CopyEdit


```
<select ...>...</select>
```

 Category सेलेक्ट करने का dropdown है: Mobile, Fashion, etc.

js

CopyEdit


```
<input ... />
```

 यूजर यहाँ टाइप करता है: जैसे "shoes" या "iPhone"।

js

CopyEdit

```
<div className="search-icon" onClick={handleSearchClick}>
  <FaMagnifyingGlass />
</div>
```


 Search बटन जिसे दबाकर result देखा जा सकता है।

js

CopyEdit

```
{suggestions.length > 0 && (
```

```
    <ul className="suggestion-box">...</ul>
  )}
```


 अगर suggestions हैं तो list में दिखते हैं — auto-suggest feature ।

Sign In / Register

js

CopyEdit

```
<div className="nav-signin">
  <Link to="/register">...</Link>
</div>
```

 "Hello, sign in" वाला पार्ट — जैसे Amazon में होता है।

Returns & Orders

js

CopyEdit

```
<div className="nav-return">...</div>
```


 Return और Order देखने का section ।

Cart

js

CopyEdit

```
<div className="nav-cart border">
  <Link to="/cart">
    <FaCartShopping />
    <span>Cart</span>
  </Link>
</div>
```

 Cart आइकन जिस पर क्लिक करके यूजर अपने टोकरी में जोड़ा हुआ सामान देख सकता है।

js

CopyEdit

```
export default Header;
```

➡ इस component को बाहर use करने के लिए export किया गया।

पूरा कोड:

```
js
CopyEdit
addToCart: (state, action) => {
  const newItem = action.payload;
  const existingItem = state.items.find(item => item.id ===
newItem.id);
  if (existingItem) {
    existingItem.quantity += 1;
  } else {
    state.items.push({ ...newItem, quantity: 1 });
  }
},
```

💬 मान ले:

तू एक ऑनलाइन समोसे की दुकान चला रहा है — **"Chatpata Samosa Store"**
तेरे पास एक **cart** (टोकरी) है: `state.items`
हर बार ग्राहक कोई समोसा जोड़ता है, यह function चलता है।

✅ अब लाइन-बाय-लाइन चले imagination में:

🟡 Line 1:

```
js
CopyEdit
addToCart: (state, action) => {
```

→ कोई customer ने "Add to Cart" बटन दबाया।
अब तू `state` (cart की टोकरी) और `action` (जो चीज ऐड हो रही है) को पकड़ता है।

Imagination:

तेरे पास एक नोट आता है:

"भाई, एक Aloo Samosa भेज दे cart में!"
यह नोट `action.payload` है।

● **Line 2:**

```
js  
CopyEdit  
const newItem = action.payload;
```

→ अब तू उस नोट से समझता है कि customer ने **कौन सा समोसा** मंगाया।

Imagination:

तू नोट को पढ़ता है और बोलता है:

"ओह! Aloo Samosa चाहिए। इसका id = 1 है।"

अब तूने इस समोसे को `newItem` नाम दे दिया।

● **Line 3:**

```
js  
CopyEdit  
const existingItem = state.items.find(item => item.id ===  
newItem.id);
```

→ अब तू देखता है कि पहले से टोकरी में यही समोसा है क्या?

Imagination:

तू टोकरी खोलता है और देखता है:

"भाई पहले से 2 Aloo Samosa रखे हैं क्या?"

अगर मिल गया — तो तू उसे `existingItem` मानता है।

● **Line 4:**

```
js
```

CopyEdit
`if (existingItem) {`

→ अगर टोकरी में **वही समोसा** पहले से पड़ा है...

Imagination:
तू बोलता है:

"हाँ भाई, पहले से रखा है Aloo Samosa, quantity 2 है।"

● Line 5:

js
CopyEdit
`existingItem.quantity += 1;`

→ तो तू उसी quantity को 1 और बढ़ा देता है।

Imagination:
तू बोलता है:

"ठीक है, एक और रख दे — अब total 3 Aloo Samosa हो गए।"

● Line 6:

js
CopyEdit
`} else {`

→ लेकिन अगर **cart** में **यह समोसा** पहले से नहीं था...

Imagination:
तू बोलता है:

"ओ भाई, यह तो पहली बार मंगाया गया है — अभी टोकरी में नहीं है।"

● Line 7:

js
CopyEdit
`state.items.push({ ...newItem, quantity: 1 });`

→ तो तू उस नए समोसे को टोकरी में डालता है और उसकी quantity = 1 रखता है।

Imagination:

तू बोलता है:

"चलो पहला Aloo Samosa टोकरी में डाल दिया — quantity: 1"

← END Line 8:

js

CopyEdit

```
},
```

→ काम पूरा — ग्राहक की टोकरी अपडेट हो गई।

👉 अब छोटा सा Example:

Cart पहले से था:

js

CopyEdit

```
state.items = [
  { id: 1, name: "Aloo Samosa", quantity: 2 }
]
```

Customer फिर से Aloo Samosa ऐड करता है।

→ तो quantity अब 3 हो जाएगी।

लेकिन अगर customer पहली बार "Paneer Samosa" ऐड करे तो:

Result:

js

CopyEdit

```
state.items = [
  { id: 1, name: "Aloo Samosa", quantity: 2 },
  { id: 2, name: "Paneer Samosa", quantity: 1 }
]
```

😎 Simple Summary:

- पहले से हो तो quantity बढ़ा दो
- नया हो तो cart में डाल दो

अगर चाहो तो इसी example को pizza, shoes, ya t-shirts से भी समझा दूँ 😊
बोलो bhai, aur मजा आया या aur examples दूँ?

4o

You said:

removeFromCart: (state, action) => { const idToRemove = action.payload; // Expecting the id
state.items = state.items.filter(item => item.id !== idToRemove); }, ab ise bhi isi style me

ChatGPT said:

Are bhai! अब पूरी style में मजेदार imagination के साथ **removeFromCart** को भी समझते हैं, जैसे तू
समोसे की दुकान चला रहा हो 😊

✅ **कोड:**

js

CopyEdit

```
removeFromCart: (state, action) => {  
  const idToRemove = action.payload;  
  state.items = state.items.filter(item => item.id !== idToRemove);  
},
```

अब लाइन-बाय-लाइन full explanation + imagination 🦋

🟡 **Line 1:**

js

CopyEdit

```
removeFromCart: (state, action) => {
```

➡ जब ग्राहक बोलता है:

"भाई, एक समोसा हटा दे मेरी टोकरी से।"
तब ये function activate होता है।

Imagination:

तेरे पास एक complaint आती है —

"भाई Paneer Samosa नहीं चाहिए अब!"

Line 2:

js

CopyEdit

```
const idToRemove = action.payload;
```

→ जो समोसा हटाना है, उसका ID इस लाइन में पकड़ लेते हैं।

Imagination:

तू complaint वाला कागज़ पढ़ता है और देखता है:

"ID 2 हटानी है? ओह! ये तो Paneer Samosa है!"

अब तू इसे नाम देता है `idToRemove = 2`

Line 3:

js

CopyEdit

```
state.items = state.items.filter(item => item.id !== idToRemove);
```

→ अब तू पूरी टोकरी (cart) से वो आइटम हटा देता है जिसकी ID = 2 है।

Imagination:

तू टोकरी उठाता है, और बोलता है:

"देखते हैं भाई — ये रहा Aloo Samosa (ID: 1) — रहने दो।

ये रहा Paneer Samosa (ID: 2) — इसे निकाल दो!"

अब सिर्फ वो समोसे बचे जो हटाए जाने वाले नहीं हैं।

Line 4:

js

CopyEdit

```
},
```

→ ग्राहक की टोकरी से सामान हटा दिया गया — काम खत्म ✓



Example:

Cart पहले था:

js

CopyEdit

```
state.items = [
  { id: 1, name: "Aloo Samosa", quantity: 2 },
  { id: 2, name: "Paneer Samosa", quantity: 1 }
];
```

Customer बोला:

"भाई Paneer Samosa निकाल दो (ID = 2)"

Resulting cart:

js

CopyEdit

```
state.items = [
  { id: 1, name: "Aloo Samosa", quantity: 2 }
];
```



Simple Version:

- Customer बोले — "इस ID वाला आइटम हटा दो"
- तू बस filter से उसे टोकरी से गायब कर देता है
- बाकियों को छोड़ देता है

भाई अब हो गया मस्त clear?

चाहे तो अगली बार "decreaseQuantity" या "clearCart" भी ऐसे ही समझा दूँ — बोले तो? 😊

4o

You said:

state.items = state.items.filter(item => item.id !== idToRemove); explain

ChatGPT said:

बिल्कुल bhai! ये लाइन बहुत important है – चल इसे बहुत आसान भाषा और मजेदार example से समझते हैं:

लाइन:

js

CopyEdit

```
state.items = state.items.filter(item => item.id !== idToRemove);
```

मतलब:

👉 टोकरी (cart) में जो भी items हैं, उनमें से जिसकी ID **idToRemove** है, उसे हटा दो। बाकी सारे items को रख लो।

Breakdown:

- `state.items` 👉 ये तेरी cart है, जिसमें सारे समोसे पड़े हैं।
 - `.filter(...)` 👉 इसका मतलब है "छाँटना" — यानी हम decide करेंगे कौन item बचेगा और कौन हटेगा।
 - `item => item.id !== idToRemove` 👉 जो item की ID **idToRemove** से अलग है, वो बच जाएगा।
-

Imagination Example:

मान लो तेरी cart में ये है:

js

CopyEdit

```
state.items = [  
  { id: 1, name: "Aloo Samosa" },  
  { id: 2, name: "Paneer Samosa" },  
  { id: 3, name: "Cheese Samosa" }  
]
```

```
];
```

और customer ने कहा:

"भाई, Paneer Samosa हटा दो।"




तब:

js

CopyEdit

```
const idToRemove = 2;
```

 अब filter चालू होता है:

- ID 1 \neq 2  बच गया
- ID 2 == 2  हटा दिया
- ID 3 \neq 2  बच गया

नया `state.items`:

js

CopyEdit

```
[  
  { id: 1, name: "Aloo Samosa" },  
  { id: 3, name: "Cheese Samosa" }  
]
```

Final Summary:

"भाई, cart में से सिर्फ वो items रहने दो जिनकी ID हटानी वाली ID से अलग हो।"

इसीलिए filter यूज़ किया — unwanted समोसे को बाहर करने के लिए! 😊

और भाई, अब ये लाइन crystal clear हो गई ना? और किसी लाइन का breakdown चाहिए क्या?

4o

You said:

increaseQuantity: (state, action) => { const id = action.payload; // Expecting the id const item = state.items.find(item => item.id === id); if (item) { item.quantity += 1; } }, // Decrease Quantity: Agar quantity 1 se zyada ho, toh reduce karo. // Agar quantity 1 hi hai, toh item ko remove karo. decreaseQuantity: (state, action) => { const id = action.payload; // Expecting the id const item = state.items.find(item => item.id === id); if (item) { if (item.quantity > 1) { item.quantity -= 1; } else { state.items = state.items.filter(item => item.id !== id); } }, }, }; ise bhi isi style me

ChatGPT said:

Are bhai! Tu toh full master ban गया Redux ke samose-wale cart mein 😊

Ab tere liye **increaseQuantity** aur **decreaseQuantity** bhi **line-by-line + full imagination example** ke साथ समझाता हूँ — जैसे तू samosa counter चला रहा हो 🧐

✅ 1. **increaseQuantity** — “भाई ek samosa aur daal do!”

पूरा कोड:

js

CopyEdit

```
increaseQuantity: (state, action) => {
  const id = action.payload;
  const item = state.items.find(item => item.id === id);
  if (item) {
    item.quantity += 1;
  }
},
```

🟡 Line 1:

js

CopyEdit

```
const id = action.payload;
```

➡ Customer ने बोला:

"भाई, Paneer Samosa ek aur chahiye!"
Tu ID पकड़ लेता है, मान ले ID = 2

● Line 2:

js

CopyEdit

```
const item = state.items.find(item => item.id === id);
```

→ अब तू अपनी टोकरी (cart) में देखता है कि ID = 2 वाला समोसा कौन है?

Imagination:

"अरे हाँ! Paneer Samosa पहले से पड़ा है, quantity = 1"

● Line 3:

js

CopyEdit

```
if (item) {
```

→ अगर समोसा टोकरी में मिल गया है...

● Line 4:

js

CopyEdit

```
item.quantity += 1;
```

→ तो quantity को 1 और बढ़ा दो

Imagination:

"ठीक है भाई! Paneer Samosa ab 2 ho gaya!"

← END Final Result:

- Cart में item पहले से है → बस quantity 1 और बढ़ा दो ✓
-

✓ 2. `decreaseQuantity` — “भाई ek samosa kam kar do... ya hata do!”

पूरा कोड:

```
js
CopyEdit
decreaseQuantity: (state, action) => {
  const id = action.payload;
  const item = state.items.find(item => item.id === id);
  if (item) {
    if (item.quantity > 1) {
      item.quantity -= 1;
    } else {
      state.items = state.items.filter(item => item.id !== id);
    }
  }
},
```

● Line 1:

```
js
CopyEdit
const id = action.payload;
```

→ Customer बोलता है:

"भाई, ek Paneer Samosa kam kar de."

तू ID पकड़ता है — मान लो ID = 2

● Line 2:

```
js
CopyEdit
const item = state.items.find(item => item.id === id);
```

→ टोकरी में से वो समोसा ढूँढता है जो ID = 2 है

● Line 3:

js

CopyEdit

```
if (item) {
```

➡ अगर समोसा मिल गया...

🔄 Line 4:

js

CopyEdit

```
if (item.quantity > 1) {  
  item.quantity -= 1;  
}
```

➡ अगर quantity 1 से ज्यादा है, तो 1 कम कर दो

Imagination:

"Haan bhai, 3 थे — ab 2 kar diye."

✗ Line 5:

js

CopyEdit

```
else {  
  state.items = state.items.filter(item => item.id !== id);  
}
```

➡ लेकिन अगर quantity सिर्फ 1 थी — तो पूरा समोसा हटा दो!

Imagination:

"Sirf 1 Paneer Samosa था — ab hata diya cart se."

📦 Example Flow:

Cart पहले था:

js

CopyEdit

```
[  
  { id: 1, name: "Aloo Samosa", quantity: 2 },  
  { id: 2, name: "Paneer Samosa", quantity: 1 }  
]
```

increaseQuantity(2)

➡ Paneer Samosa → quantity: 2

decreaseQuantity(2)

➡ Paneer Samosa → quantity: 1 → फिर हट जाएगा!



Simple Summary:

Action

Kya Hota Hai

increaseQuantity Cart में वो समोसा पहले से है → quantity +1
ity

decreaseQuantity अगर quantity > 1 → -1 करो, वरना पूरा item हटा दो
ity